

CHMS: A Boundary-Gated Computational Habitat Framework for Prime-State Cellular Simulation

A computational sequel to *The Coherence Lattice: A Foundational Harmonic-Geometric and Bioelectric Framework for DNA, RNA, Telomeres, and Biological Regeneration*

Technical White Paper / Journal-Style Research Monograph

Elisha B. Parker

April 2026

Keywords: computational biology; habitat manifold; bioelectricity; telomeres; boundary circuits; cellular simulation; gating; state-space modeling; provenance-aware software; regenerative systems; multiscale simulation; oscillatory environments

Abstract

This paper presents **CHMS — Cellular Habitat Manifold Simulator** as a computational sequel to the Coherence Lattice framework. The foundational paper argued that biological stability may depend on a deeper multiscale order state joining electrical patterning, geometry, accessibility, boundary integrity, and bounded regeneration, and explicitly proposed AI-mediated multiscale simulation as the correct intermediate proving ground between theory and laboratory intervention.

The present work operationalizes that proposal. CHMS does not attempt a full molecular duplication of life and does not claim clinical proof of rejuvenation. Instead, it encodes a single virtual cell inside a nested habitat manifold whose environment, shell boundaries, gates, disturbance classes, and recovery trajectories can be perturbed, inspected, exported, and compared over time. The simulator uses a mixed scaffold of directly grounded biological anchors, composite proxies, and latent readout metrics. Its purpose is to test whether better-structured environments make lawful cellular behavior easier to sustain and recover under bounded constraints.

The core claim of this paper is that the Coherence Lattice framework can be translated into a functioning simulation instrument without losing scientific discipline. CHMS demonstrates that nested boundary architecture, gating behavior, disturbance, and recovery can be rendered as a mathematically explicit, code-executable, visually inspectable state-space system. The project's central systems insight is that the simulator does not converge toward one universal voltage or one universal frequency. Instead, its deepest recurring pattern is **nested boundary-conditioned electrochemical admissibility**: gradients must be held, boundaries must preserve identity, access must be regulated, and lawful transitions must occur inside bounded corridors. This formulation is especially consistent with the foundational paper's treatment of telomeres as boundary circuits performing protection, regulated access, structural gating, and bounded renewal.

This paper therefore focuses on the computational instrument itself: what it is, what it represents, what it does not represent, how its mathematics are translated into code, what the current working build already demonstrates, and what horizons now open from this architecture. Only in the final section does the paper address the oscillatory and audible extension, placing it explicitly as a secondary mirror of the simulator's canonical oscillatory architecture rather than as the primary biological claim. In this form, CHMS is offered as a disciplined preclinical simulation environment through which the Coherence Lattice thesis may be constrained, refined, and prepared for stricter empirical nomination.

1. Introduction

1.1 Why a second paper is necessary

The foundational Coherence Lattice paper established a broad first-principles thesis: that living systems may be better understood as electrically active, mechanically responsive, and geometrically organized matter, and that aging and regeneration may need to be reinterpreted as movement through a deeper order architecture rather than through chemistry alone. It also made a crucial methodological move: it stated that any serious development of this framework would require a computational proving ground capable of encoding multiscale variables before laboratory validation.

That proving ground now exists.

CHMS was built inside this thread as the first operational computational body of the framework. It emerged not as a generic visualization package, not as an animation exercise, and not as a “healing-frequency” app, but as a research instrument designed to ask a highly specific question:

What kind of environment makes lawful cellular behavior easiest to resume and sustain?

This paper is therefore not a rehash of the foundational biology paper. It is Part II in the strictest sense: the transition from theory into instrument.

1.2 The shift from command model to habitat model

One of the deepest conceptual clarifications in this thread was that the simulator should not be built as a direct command system. The aim is not to issue instructions to the cell. The aim is to construct a **habitat manifold** that preserves the conditions under which the cell’s own lawful intelligence becomes easiest to express. This is perfectly aligned with the foundational paper’s conceptual dynamics equation, where damage and noise oppose endogenous restoration and structured external ordering input, and where prime-state biology is defined not as perfection, but as a bounded regime in which errors can still be absorbed or corrected without global destabilization.

This distinction matters enormously. It changes the tone of the whole project:

- not “what frequency forces the cell to heal?”
- but “what support architecture makes correct behavior easiest to resume?”

The present paper treats that shift as one of the central theoretical advances of CHMS.

1.3 The central contribution of CHMS

The contribution of CHMS is not that all of biology has now been solved in silico. That would be false. The contribution is that a first-principles theory of biological order has now been made:

- **mathematically explicit**
- **code-executable**
- **visually inspectable**
- **provenance-aware**
- **exportable and comparable**

- **calibratable without rewriting its core doctrine**

That is already significant. It means the Coherence Lattice is no longer only a manifesto. It has a working computational body.

2. Continuity with the Foundational Framework

2.1 What CHMS inherits directly

CHMS inherits several propositions directly from *The Coherence Lattice*.

First, the framework is multicomponent. The foundational paper explicitly defines a biological-state vector including electrical state, geometric state, access-state organization, boundary integrity state, regenerative competence, epigenetic/regulatory state, metabolic-redox state, and telomeric state.

Second, it treats coherence as a **latent composite variable** rather than a directly isolated object. The Coherence Lattice is useful only if it can be inferred from measurable observables and structured subdomains.

Third, the foundational paper defines prime-state biology and degraded-state biology as **bounded regions in state space**, not as single-number states. Prime state is a regime in which structural relations, access-state control, electrical patterning, and regenerative competence remain inside admissible bounds, while degraded state is a regime in which those bounds erode and noise propagates more easily.

Fourth, it explicitly proposes an **AI-mediated multiscale simulation environment** as the intermediate proving ground between conceptual theory and laboratory intervention. The purpose of that simulator is not to replace biology, but to determine whether a sufficiently constrained virtual environment can reproduce lawful relationships among structure, electrical state, boundary integrity, and oscillatory input in ways that generate falsifiable predictions.

CHMS is the direct realization of that proposal.

2.2 Telomeres as proof arena

CHMS also inherits a very specific telomere doctrine. In the foundational work, telomeres are not treated merely as repetitive chromosome ends that

shorten with age. They are explicitly treated as **boundary circuits** performing four simultaneous tasks: boundary protection, regulated access, structural gating, and bounded renewal.

That doctrine became one of the core architectural principles of CHMS. The simulator does not merely include telomeres as another state variable; it treats telomeric state as the sharpest available readout of boundary-conditioned biological order.

2.3 Measurement logic inherited from Chapter 9

The foundational paper's Chapter 9 argued that a new theory of biological order requires a new measurement logic, one built around distributions, boundary function, access-state, electrical coherence, and resilience-to-return rather than around mean-only biomarkers. It also explicitly proposed the need for simulation-calibrated coherence observables and multiscale state-space reconstruction.

This is exactly the logic that CHMS operationalizes through:

- prime-state versus degraded-state runs
- disturbance and recovery trajectories
- frame-linked readouts
- provenance-aware live metrics
- shell-specific inspection
- exportable traces and stage configurations

The project therefore remains continuous with the foundational white paper not only conceptually, but methodologically.

3. Discovery and Scope

3.1 The strongest truthful discovery statement

The strongest truthful statement that can now be made is this:

CHMS demonstrates that the Coherence Lattice hypothesis can be translated into a functioning computational habitat simulator in which nested boundary architecture, gating behavior, disturbance, and recovery can be represented, inspected, exported, and compared under staged environmental conditions.

This is already a major achievement.

3.2 What has actually been achieved

Within the current build, the following have already been achieved in practice:

- a local browser-based simulator launches and runs
- staged environments can be loaded and executed
- one virtual cell is represented inside a nested shell architecture
- prime, degraded, intermediate, and recovery states can be inspected across time
- morphology playback is linked to underlying state rather than decorative motion
- quantitative traces and visual state changes are linked to the same underlying run logic
- different disturbance classes produce distinct signatures
- a stage editor, provenance panel, disclaimer panel, export logic, camera focus controls, per-shell visibility, and frame-linked inspection surfaces have been integrated additively without destabilizing the solver core

3.3 What has not been achieved

To keep the scope exact, CHMS does **not** presently prove:

- full biological duplication
- clinical rejuvenation
- that virtual success is equivalent to wet-lab truth
- that the simulator's present scalar readouts are already biologically validated
- that one master variable or one master frequency governs the whole system

The simulator is preclinical by design. This restraint is not a weakness; it is what preserves the project's scientific usefulness. The foundational paper itself insisted that simulation is an intermediate testbed whose function is to determine whether constrained virtual environments can reproduce lawful relations strongly enough to justify experimental follow-up.

3.4 Why this is still monumental

The fact that CHMS remains preclinical does not make it trivial. The real achievement is that a first-principles theory of biological order now has a computational body capable of:

- declaring its variables

- encoding its constraints
- exposing its assumptions
- showing its failure modes
- supporting calibration
- yielding exportable comparison runs
- preparing later nomination to empirical work

That is exactly the threshold the foundational paper identified as the transition from conceptual manifesto to technical program.

4. The Unifying Pattern Revealed by the Simulator

4.1 Not one universal voltage, not one universal tone

One of the most important outcomes of the thread was the discovery that the simulator does not collapse cleanly into one universal voltage or one universal tone. That temptation had to be resisted.

Instead, the deepest pattern that emerged was:

nested boundary-conditioned electrochemical admissibility

That phrase captures the recurring logic across the system:

- gradients are maintained
- boundaries preserve identity
- gates regulate access
- lawful transitions occur only inside admissible corridors
- degraded systems lose the ability to preserve those relations cleanly

4.2 Why this matters more than a scalar answer

This insight is more useful than a single-number answer because it scales.

At the outer environment level, the question is whether the chamber preserves low-noise, supportive, bounded conditions.

At the membrane level, the question is whether gradients and access remain lawful.

At the nuclear level, the question is whether the genomic environment remains selectively accessible.

At the telomeric level, the question is whether the end-state remains protected, gated, and renewably bounded.

This means the simulator's discovery is architectural, not merely numerical.

4.3 Telomeres as the sharpest instance

The reason telomeres remain central is that they compress the whole pattern into one high-sensitivity arena. The foundational paper's definition of telomeres as boundary circuits becomes computationally powerful in CHMS because it provides a model of what the entire system is doing:

- boundary protection
- regulated access
- structural gating
- bounded renewal

CHMS generalizes that telomeric logic outward into the habitat manifold itself.

5. Research-Grounded Variables and Their Translation into CHMS

5.1 Variable classes

To remain scientifically honest, CHMS classifies its variables into three classes:

1. **directly grounded variables**
2. **composite proxies**
3. **latent or derived readouts**

This is not cosmetic. It is a core doctrine of the simulator and is reflected in its provenance surfaces and export logic.

5.2 Directly grounded anchors

The directly grounded anchors used in CHMS include:

- a nominal **pH environment corridor** centered at 7.4 for the outer environment layer
- a practical **membrane-potential reference corridor** centered near -60 mV for the cell electrical context
- a **DNA-repair redox anchor** around the DNA-bound [4Fe4S] repair-enzyme regime near $+80$ mV vs NHE

- a **primase operational redox window** represented as a simplified two-state regime around approximately +80 mV and +160 mV vs NHE
- an explicitly non-scalar **telomeric boundary context** derived from the foundational paper's telomere-state formulation rather than from a claimed telomere voltage law

These anchors are not being claimed as final universal constants. They function as grounded corridor references for simulation.

5.3 Why telomeric state stays composite

One of the strongest and most accurate design choices in CHMS is that telomeric state is not represented as a single canonical voltage. Instead, it is modeled as a composite of:

- protection state
- accessibility state
- chromatin-structural state
- regulatory occupancy
- ion/topology support
- bounded renewal compatibility

This choice is not merely defensible; it is demanded by the foundational paper's telomere doctrine.

5.4 The role of composite proxies

Because not all layers of biological order present themselves as one directly measurable scalar, CHMS uses structured proxies such as:

- support score
- telomere ion/topology score
- shell integrity
- boundary integrity
- gating quality

These are not “made up” in the careless sense. They are **declared proxy surfaces** that bridge grounded variables to inspectable system behavior.

5.5 Latent and derived readouts

Finally, CHMS includes derived readouts such as:

- core coherence (C)
- overall coherence

- harmonic complexity (H)
- recovery capacity (R)
- breakdown risk (B)

A critical refinement in the current working build is that `overall_coherence` was introduced as a **boundary-weighted research-facing readout only**, while raw `state.C` remained untouched inside the solver. This preserved legacy stability while improving interpretive fidelity under visible shell degradation.

This is an important model of how future refinements should proceed: add interpretive clarity before altering dynamics.

6. Mathematical Architecture

6.1 Environmental state vector

CHMS models the habitat manifold through an environmental state vector of the form:

$$\begin{bmatrix} E(t) = [\text{pH}, I, F, N, B_s, A_p] \end{bmatrix}$$

where:

- (pH) = proton/redox context
- (I) = ion/topology support
- (F) = external field order
- (N) = noise load
- (B_s) = boundary support
- (A_p) = access permissivity

This is the chamber logic. These values define what the environment makes easy or hard.

6.2 Cellular state vector

The virtual cell is modeled through:

$$\begin{bmatrix} X(t) = [V, G, M, T, A] \end{bmatrix}$$

where:

- (V) = bioelectric state
- (G) = geometric/structural state
- (M) = metabolic-redox state
- (T) = telomeric boundary state
- (A) = access-state quality

This is a computational specialization of the broader state-vector logic in the foundational paper.

6.3 Dynamic update doctrine

The simulator follows the conceptual directional rule:

$$\begin{bmatrix} x_{t+1} = x_t + \text{repair} + \text{support} - \text{damage} - \text{noise} \end{bmatrix}$$

This is not treated as a fitted law of nature. It is a formal simulation scaffold that lets the theory declare how support, repair, damage, and disorder relate.

6.4 Distance from prime state

The foundational paper explicitly defined prime and degraded biology as movement through a bounded manifold. CHMS therefore treats health not as one number, but as **proximity to a prime-state basin** rather than to a point.

This is why the simulator supports:

- prime-state presets
- disturbed-state presets
- staged degradation
- recovery ramps
- comparison between trajectories rather than single snapshots

6.5 Gates as mathematical and visual objects

CHMS does not merely talk about gating conceptually. It models at least three gate classes:

- membrane gate
- repair gate

- telomeric maintenance gate

These gate values affect not only state updates but also the visual and interpretive surfaces of the app. This is one of the places where the computational sequel most clearly fulfills the promise of the foundational paper: it converts an abstract access-state thesis into runnable architecture.

7. The Virtual Environment as Research Instrument

7.1 Why the UI matters scientifically

A major insight of this thread was that the UI is not secondary. The interface is part of the instrument.

If the simulator is to support real research reasoning, the user must be able to:

- define staged environments
- inspect the resulting state
- compare runs apples-to-apples
- see what is grounded versus proxy versus latent
- isolate shell layers
- inspect breakdown and recovery by frame
- export those results cleanly

That is why the implementation passes did not merely wire up the app to run. They progressively transformed it into a **provenance-aware inspection instrument**.

7.2 Stage editor and timeline logic

The simulator now includes a stage editor built as a Streamlit-native table-plus-visual workflow rather than a custom drag component. This was an explicit design decision made to keep v1 legacy-safe and implementation-ready. Stages include environment fields, disturbance class, intensity, transition behavior, duration, and ordering, and they round-trip through export schemas such as `stages.csv` and `trace.csv`.

This is not a generic configuration panel. It is the place where the habitat manifold becomes experimentally legible.

7.3 Provenance as an epistemic surface

CHMS includes a provenance registry whose row shape includes:

- group
- label
- symbol
- category
- meaning
- adjustable state
- value source
- current value

This means the app does not merely present a score. It presents a score **and** its status inside the knowledge architecture of the simulator. This is a major strength of the project. It prevents grounded values, proxies, and latent readouts from collapsing into one indistinguishable UI layer.

7.4 Inspection panel and dual context

A later refinement pass sharpened the instrument further by explicitly separating **Run Context** from **Inspection Context**, and by creating a dedicated right-side panel that remains tied to the selected frame. That panel includes:

- inspection context
- frame metrics
- shell state
- provenance snapshot

This refinement is central because it lets the user move between:

- what the viewer is showing
- what the selected frame metrics are
- what stage produced the frame
- which values are grounded, proxy, or derived

This is the kind of clarity a true research instrument requires.

7.5 Per-shell visibility and camera focus

The project also added per-shell visibility controls and camera focus targets. Shell visibility remains visual-only and does not alter the simulation logic. This preserves the difference between reading the system and changing the system. Camera focus, meanwhile, allows the user to move quickly between

full architecture and specific shells without breaking the nested architecture logic.

8. What the Current Working Build Proves

8.1 The simulator expresses state, not just motion

The current screenshots and refinement passes show that CHMS is not simply animating pleasing movement. It is expressing state.

This is a fundamental distinction. The morphology playback layer already demonstrates:

- healthy initial state
- intermediate degradation
- later breakdown
- recovery under improved conditions

This means the app is already doing what the computational sequel most needed to do: showing that the theory can be represented as time-evolving lawful state rather than as static doctrine.

8.2 Disturbance classes are not generic collapse modes

The current working build distinguishes between at least different disturbance signatures such as oxidative load and boundary weakening. This matters because it means CHMS is not simply mapping all degradation into one collapse channel. It is showing that different environmental insults produce different system trajectories.

This is essential if the simulator is to be useful later for hypothesis nomination.

8.3 Readout polish proves the legacy-safe refinement strategy

The boundary-weighted `overall_coherence` pass is especially important as a programming result. It proved that the working core could be sharpened interpretively without destabilizing the solver. The model preserved raw `state.C`, added a boundary-sensitive readout layer, extended the export schema, and synchronized the inspection surfaces — all without dynamics regression.

This is not merely a UI success. It demonstrates a methodological success: **CHMS can be refined additively without violating its own doctrine.**

8.4 The software itself is evidence of feasibility

The project's programming choices matter because they prove a different kind of point than the biological theory alone. They show that the framework is implementable in a disciplined way:

- one protected implementation root
- one local Python doctrine
- a stable project-local venv
- a legacy-safe Streamlit/Plotly stack
- narrow helper modules rather than unnecessary rewrites
- additive feature passes rather than architecture churn

In other words, the software engineering path itself is proof that the concept can survive operational constraint.

9. Programming, Architecture, and What the Build Itself Demonstrates

9.1 Why the code matters philosophically

The code is not just a delivery mechanism. It is a philosophical filter. A theory can sound elegant until it is forced into variables, file boundaries, interfaces, schemas, and update rules. CHMS has now passed through that filter.

The build forced the project to specify:

- what the variables actually are
- which are grounded
- which are proxy
- which are latent
- what the state update law is
- what the UI must reveal
- what must remain protected legacy
- what counts as a meaningful test

That is one of the strongest achievements of the project. The code has disciplined the theory.

9.2 Legacy-safe modular coding as scientific method

A remarkable byproduct of this thread is that software-architecture discipline became part of the scientific method of the project.

The commitment to:

- preserve the working core
- add only narrow helper modules
- keep v1 inside a local browser shell
- separate readout from dynamics
- avoid destabilizing rewrites

was not just an engineering preference. It became a methodological necessity. It mirrors the paper's own insistence on bounded restoration rather than indiscriminate intervention.

The software process, in this sense, became homologous to the biological thesis.

9.3 Why this matters for future forks

Because CHMS is now organized as a working instrument with explicit doctrinal boundaries, it is forkable in a principled way. Future branches can extend the framework without invalidating the core. That makes the project durable.

10. Horizons Opened by the Current Build

10.1 Immediate horizon: calibration and stricter comparison

The first horizon is not speculative at all. It is already open.

CHMS can now be pushed further in:

- metric calibration
- shell-state fidelity
- boundary sensitivity
- readout refinement
- run-to-run comparison
- stronger export analysis
- richer prime/degraded manifold exploration

This is the next discipline layer of the current instrument.

10.2 Second horizon: richer disturbance families

The simulator can also evolve into a more nuanced disturbance-and-recovery environment. Future forks can introduce:

- more complex noise classes
- asynchronous boundary failures
- compound disturbance scenarios
- shell-specific stressors
- adaptive recovery ramps
- resilience-to-return measurements consistent with the foundational paper's proposed metrics

10.3 Third horizon: richer measurement nomination

Because the simulator now supports grounded, proxy, and latent layers, it can become a strong nomination engine for later measurement architectures:

- which shell variables deserve stronger future biological anchoring
- which distributions are most informative
- which boundary failures create the clearest system signatures
- which readout combinations best distinguish degraded-state from recovering-state runs

This is one of the most important future forks because it links directly back to the foundational paper's Chapter 9 measurement program.

10.4 Fourth horizon: multiscale expansion

The current build is centered on one virtual cell in one habitat manifold. But the architecture is now clear enough that future forks may explore:

- multicell networks
- coupled habitat manifolds
- tissue-like shell interactions
- intercellular electrical coupling layers
- chromatin-state microdomains
- richer telomeric submodels

These are not yet part of the current truthful claim. They are genuine future forks now opened by the working architecture.

10.5 Fifth horizon: experimental handoff

The most ambitious but still disciplined future fork is experimental handoff. CHMS can now function as a preclinical nomination engine for:

- which structured environments should be tested
- which disturbance classes matter most
- which oscillatory or electrical perturbations are worth exploration
- which boundary variables should be prioritized in wet-lab work

This is where the simulator most clearly fulfills the role proposed by Chapter 10 of the foundational paper.

11. Potential Future Forks

11.1 Fork A — Research-grade CHMS

This fork would prioritize:

- stronger provenance
- better calibration
- richer export formats
- formal metric validation
- reusable experiment packs
- prime/degraded manifold libraries

This would move CHMS toward a publishable research instrument.

11.2 Fork B — Educational CHMS

This fork would preserve the theory but simplify the interface for broader interpretability:

- guided tours
- visual teaching overlays
- shell explanations
- disturbance walkthroughs
- simplified stage recipes

This could make the framework legible to wider audiences without diluting the core.

11.3 Fork C — Wet-lab companion planning tool

This fork would shift CHMS toward experimental design:

- hypothesis nomination dashboards
- variable-to-assay mapping
- candidate perturbation planners
- simulation-to-measurement alignment tools

11.4 Fork D — Oscillatory mirror and sonification

This fork remains secondary and is addressed last by design. Its purpose would be not to replace the simulator, but to mirror its canonical oscillatory architecture in structured auditory or signal terms.

This is important, but it must remain downstream of the virtual environment and computational doctrine.

12. The Audible Extension, Properly Placed Last

12.1 Why it comes last

The audible extension belongs last because the primary research contribution of this thread is **the virtual environment and its executable state architecture**, not the audio. The simulator itself is the discovery body. The audible layer is an extension of that body, not its foundation.

12.2 The strict doctrine

The doctrine established in this thread is:

- static biological voltages do **not** become pitch directly
- only explicitly oscillatory CHMS processes may be translated 1:1 in Hz
- static biological state anchors remain canonical in their own domains
- an audio layer may mirror canonical oscillator families without pretending volts are already notes

This yields the correct pipeline:

[
\text{biology anchors} \rightarrow \text{state architecture} \rightarrow \

text{oscillatory architecture} \rightarrow \text{audible mirror}
]

12.3 What the audible work has already shown

The thread has already explored a doctrine-faithful oscillatory family and created exact renders and MTS MIDI variants built around a rooted hierarchy including a 16 Hz conductive-environment base and higher layers at 64 Hz, 128 Hz, 256 Hz, and 512 Hz. These remain secondary artifacts, but they demonstrate that CHMS can support not only visual but also oscillatory mirroring of its own architecture.

12.4 Why the audible layer still matters

The audible extension matters because it offers:

- a second inspection modality
- a structural mirror of hierarchy and gating
- a way of hearing staged entry and return to root
- a possible future branch for oscillatory-environment exploration

But it remains secondary. The major discovery of Part II is the **computational habitat instrument** itself.

13. Conclusion

This paper has presented CHMS as the computational sequel to *The Coherence Lattice*. The foundational paper established the conceptual substrate: biological order as a multiscale harmonic-geometric and bioelectric architecture, telomeres as boundary circuits, and simulation as the necessary intermediate proving ground between theory and empirical intervention. CHMS now fulfills that computational role. It does not prove the entire biological framework in vivo, and it does not replace laboratory biology. What it does accomplish is more rigorous and more durable: it converts a first-principles theory of biological order into a mathematically explicit, code-executable, visually inspectable, provenance-aware simulation instrument.

That is already a major threshold crossing.

The work of Part II is therefore not to claim final closure. It is to state clearly what has already been achieved:

- a habitat-manifold model rather than a command model
- a working virtual cell inside a nested shell architecture
- research-grounded anchors translated into explicit variable classes
- dynamic gates, disturbances, and recovery paths
- a UI that behaves as a research instrument
- additive calibration without legacy destabilization
- clear horizons for calibration, measurement, multiscale expansion, and experimental handoff
- an audible mirror doctrine placed correctly downstream of the simulator rather than above it

If the first paper established the possibility of a coherence architecture, this paper establishes its first operational body.

CHMS is that body.

Appendix A. Core CHMS Variable Classes

A.1 Directly grounded variables

- pH environment corridor
- membrane-potential corridor anchor
- repair-redox anchor
- primase operational redox window

A.2 Composite proxies

- support score
- boundary integrity
- telomere ion/topology score
- shell state descriptors

A.3 Latent or derived readouts

- core coherence (C)
 - overall coherence
 - harmonic complexity (H)
 - recovery capacity (R)
 - breakdown risk (B)
-

Appendix B. Core Implemented UI and Data Surfaces

- local Streamlit shell
 - Plotly nested-shell viewer
 - stage editor
 - provenance panel
 - disclaimer panel
 - export panel
 - camera focus controls
 - per-shell visibility controls
 - dual-context labeling
 - frame-linked inspection/readout panel
 - stages.csv
 - trace.csv
 - JSON run payload
 - provenance snapshot export
-

Appendix C. Research Questions Now Open

1. Which environmental support variables contribute most strongly to resilience-to-return?
2. Which shell-level degradation patterns best predict broader system collapse?
3. Which composite proxies deserve stronger empirical grounding next?
4. Which staged perturbations generate the clearest prime/degraded separation?
5. Which future wet-lab measurements best align with the simulator's most informative observables?
6. Which canonical oscillatory architectures, if any, merit future bounded exploration as secondary mirrors of the simulator's state-space logic?